Computer Science Department
cs.salemstate.edu

Note: last offering of this course was Spring 2017

CSC 311 Object Orientation and Problem Solving Strategies **4 cr.**

Instructor: TBA **Office:** location **Phone:** (978) 542-extension
email: TBA@salemstate.edu **Office Hours:** days and times

Section	Time	Room	Final Exam
nn	days and times	location	date and time
Lnn	days and times	location	

Catalog description:

This course presents a basic treatment of the use of toolkits, frameworks, and design patterns in object-oriented design and programming. The concepts of composition, component reuse, inheritance, and parametrization (templates) are studied and used to develop problem-solving strategies, which are then implemented in one or more current object-oriented languages. (Consult the instructor for the language(s) to be used.) Three lecture hours and three hours of scheduled laboratory per week, plus extensive programming work outside of class.

Prerequisite: CSC 260.

Goals:

CG01: to present a unified discussion of the important ideas and techniques involved in the use of design patterns and frameworks in object-oriented software design;
CG02: to guide the student through one or more large-scale projects employing these tools;
CG03: to provide additional experience in programming in an object-oriented language (normally a language other than that students already know).

Objectives:

Upon completion of this course, the student will have demonstrated the ability to:

CO01: explain the concepts of framework and design pattern and the differences between them;
CO02: name and explain the use of several of the most common design patterns, using appropriate UML diagrams;
CO03: understand and employ the various types of reuse in object-oriented design;
CO04: understand and use a variety of components from one or more standard software libraries;
CO05: understand and use the concept of templates in object-oriented design;
CO06: design and implement a solution to a large-scale problem using object-oriented tools, and provide appropriate documentation for the solution.

Program Outcome vs. Course Objectives matrix

Program Objective (condensed form)	CO01	CO02	CO03	CO04	CO05	CO06
PO-A: apply knowledge of computing and math	✓	✓	✓	✓	✓	✓
PO-B: analyze a problem and define its computing requirements	✓	✓	✓	✓	✓	✓
PO-C: design, implement and evaluate applications			✓	✓	✓	✓
PO-D: function effectively in teams to accomplish a common goal						
PO-E: professional, ethical, and social responsibilities						
PO-F: communicate effectively with a range of audiences						✓

Program Objective (condensed form)	CO01	CO02	CO03	CO04	CO05	CO06
PO-G: local and global impact of computing on people and society						
PO-H: need for continuing professional development						
PO-I: use current techniques, skills, and tools	✓	✓	✓	✓	✓	✓
PO-J: apply theory and principles to model and design systems	✓	✓	✓		✓	✓
PO-K: apply design and development principles in constructing software						✓
note - full statements of the Program Outcomes (program objectives) for the Computer Science Major can be found in the document *Computer Science Major Program Educational Objectives and Program Outcomes* on the Assessment page of the Computer Science Major (cs.salemstate.edu)						

Topics:

- Programming language concepts
  - programming paradigms (two hours)
    - procedural
    - object-oriented
    - functional
    - declarative
    - multi-paradigm
    - emerging and/or specialized paradigms
  - type systems (one hour)
    - "strongly-typed" vs "weakly-typed" vs. "type safe"
    - static (compile-time) type checking
    - dynamic (run-time) type checking
  - memory allocation and management (one hour)
    - static vs. dynamic
    - direct vs. indirect
- The engineering of software
- Review of object-oriented concepts and terminology (and the associated UML diagrams): **PL6(3)**, **PF1(2)**
  - behavior and state
  - instances and classes
  - coupling and cohesion
  - interface and implementation
- Reuse
  - composition
  - aggregation
  - inheritance
- Problem Solving **SE1(4)**, **SE6(2)**, **PF5(2)**
  - the role of top down design and stepwise refinement
  - steps, cases, divide and conquer, and commonality
- Creating software components in an object oriented language (such as C++ or Java) **PL4(2)**,**PL6(7)**
  - classes and objects
  - streams
  - containers and iterators
- Component reuse **PL5(2)**
  - templates
  - inheritance
- Frameworks and design patterns
  - the nature of design patterns
  - survey of some common design patterns
    - Abstract Factory, Adapter, Mediator, and others
  - design patterns vs. frameworks
- Problem-solving strategies **PF3(6)**, **AL2(4)**, **AL3(4)**
  - breadth first algorithms
  - backtracking algorithms

- Use of libraries **SE2(2)**
  - the Standard Library in C++
  - Microsoft Foundation Class library
  - Java Foundation Class library

Case Studies and / or Laboratory Exercises:

The following topics are typical of those used as class examples and programming assignments:

- strings and string processing; pattern matching
- rational number and complex number classes
- large precision integers
- an inventory control system or a payroll system
- a simple windowing system
- bit vectors and applications
- knight's tour problem
- graph applications

Programming Assignments: There will be 3 to 5 substantial programming assignments (possibly one large project divided into progressive stages) in which students will be required to implement appropriate design components. Considerable attention will be paid to design considerations and the selection of appropriate tools (control structures, design patterns, algorithms, and software library facilities).

All programs must conform to departmental guidelines for design and implementation, and laboratory reports must conform to the written guidelines supplied by the instructor. Regardless of numeric average or grades on individual assignments or examinations, a student will not be eligible for a passing grade in the course unless he or she has submitted a lab report for every assignment within the timeframe specified by the instructor.

Laboratory exercises: There will be short programming and design exercises to be completed during weekly scheduled laboratory sessions. Laboratory exercises will concentrate on language syntax, programming style, and familiarization with the contents of the Standard C++ Library

Exams and quizzes: There will be one mid-term exam and a comprehensive written two-hour final examination.

The course grade will be determined using the following approximate weights: laboratory exercises - 20%, programming assignments – 40%, examinations (midterm and final) -30%, written homework - 10%.

Course Objective / Assessment Mechanism matrix

	Written Homework	Laboratory Exercises	Programming Projects	Examinations
CO01	✓			✓
CO02	✓			✓
CO03			✓	✓
CO04	✓	✓	✓	✓
CO05	✓	✓	✓	✓
CO06			✓	

Bibliography:

Cormen, Thomas H.; Leiserson, Charles H.; Rivest, Ronald L.; Stein, Clifford. **Introduction to Algorithms**. **Third Edition.** The MIT Press, 2009.

Deitel, Harvey M.; Deitel. Paul J. **C++ How to Program: Early Objects Version. Eighth Edition.** Prentice Hall, 2012.

Fowler, Martin; with Scott, Kendall. **UML Distilled: A Brief Guide to the Standard Object Modeling Language**. **Third Edition.** Addison-Wesley, 2003.

Friedman, Frank L.; Koffman, Elliot B. **Problem Solving, Abstraction, and Design using C++. Sixth Edition.** Addison-Wesley, 2011.

Gaddis, Tony**;** Walters, Judy; Muganda, Godfrey. **Starting Out with C++: Early Objects. Seventh Edition.** Addison-Wesley, 2011.
Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995.
Horowitz, Ellis; Sahni, Sartaj; Rajasekaran, Sanguthevar. **Computer Algorithms/C++. Second Edition.** Silicon Press, 2008.
Horstmann, Cay S. **C++ for Everyone. Second Edition.** Wiley, 2011.
Horstmann, Cay S.; Budd Timothy A. **Big C++. Second Edition.** Wiley, 2009.
Main, Michael; Savitch, Walter. **Data Structures and Other Objects Using C++. Fourth Edition.** Addison-Wesley, 2011.
Musser, David R.; Saini, Atul; Derge, Gillmer J. **STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library**. Addison-Wesley, 2009.
Oestereich, Bernd. **Developing Software with UML: Object-Oriented Analysis and Design in Practice. Second Edition.** Addison-Wesley, 2002.
Prata, Stephen. **C++ Primer Plus. Fifth Edition.** Sams, 2004.
Savitch, Walter J. **Problem Solving with C++. Seventh Edition.** Addison-Wesley, 2008.
Schildt , Herbert. **C++ Programming Cookbook**. McGraw-Hill, 2008.
Schildt , Herbert. C++. **The Complete Reference, 4th Edition.** McGraw-Hill, 2008.
Shalloway, Alan; Trott, James R. **Design Patterns Explained: A New Perspective on Object-Oriented Design. Second Edition.** Addison-Wesley, 2004.
Weiss, Mark Allen. **Data Structures and Algorithm Analysis in C++. Third Edition.** Addison-Wesley, 2007.

Academic Integrity Statement:

"Salem State University assumes that all students come to the University with serious educational intent and expects them to be mature, responsible individuals who will exhibit high standards of honesty and personal conduct in their academic life. All forms of academic dishonesty are considered to be serious offences against the University community. The University will apply sanctions when student conduct interferes with the University primary responsibility of ensuring its educational objectives." Consult the University catalog for further details on Academic Integrity Regulations and, in particular, the University definition of academic dishonesty.

The Academic Integrity Policy and Regulations can be found in the University Catalog and on the University website (http://catalog.salemstate.edu/content.php?catoid=13&navoid=1295#Academic_Integrity). The formal regulations are extensive and detailed - familiarize yourself with them if you have not previously done so. A concise summary of and direct quote from the regulations: "Materials (written or otherwise) submitted to fulfill academic requirements must represent a student's own efforts". *Submission of other's work as one's own without proper attribution is in direct violation of the University's Policy* and will be dealt with according to the University's formal Procedures. *Copying without attribution is considered cheating in an academic environment - simply put,* ***do not do it!***

University-Declared Critical Emergency Statement:

In the event of a university-declared emergency, Salem State University reserves the right to alter this course plan. Students should refer to www.salemstate.edu for further information and updates. The course attendance policy stays in effect until there is a university-declared critical emergency.

In the event of an emergency, please refer to the alternative educational plans for this course, which will be distributed via standing class communication protocols. Students should review the plans and act accordingly. Any required material that may be necessary will have been previously distributed to students electronically or will be made available as needed via email and/or Internet access.

Equal Access Statement:

"Salem State University is committed to providing equal access to the educational experience for all students in compliance with Section 504 of The Rehabilitation Act and The Americans with Disabilities Act and to providing all reasonable academic accommodations, aids and adjustments. **Any student who has a documented disability requiring an accommodation, aid or adjustment should speak with the instructor immediately.** Students with Disabilities who have not previously done so should provide documentation to and schedule an appointment with the Office for Students with Disabilities and obtain appropriate services."

Note: This syllabus represents the intended structure of the course for the semester. If changes are necessary, students will be notified in writing and via email.